FIG. 1

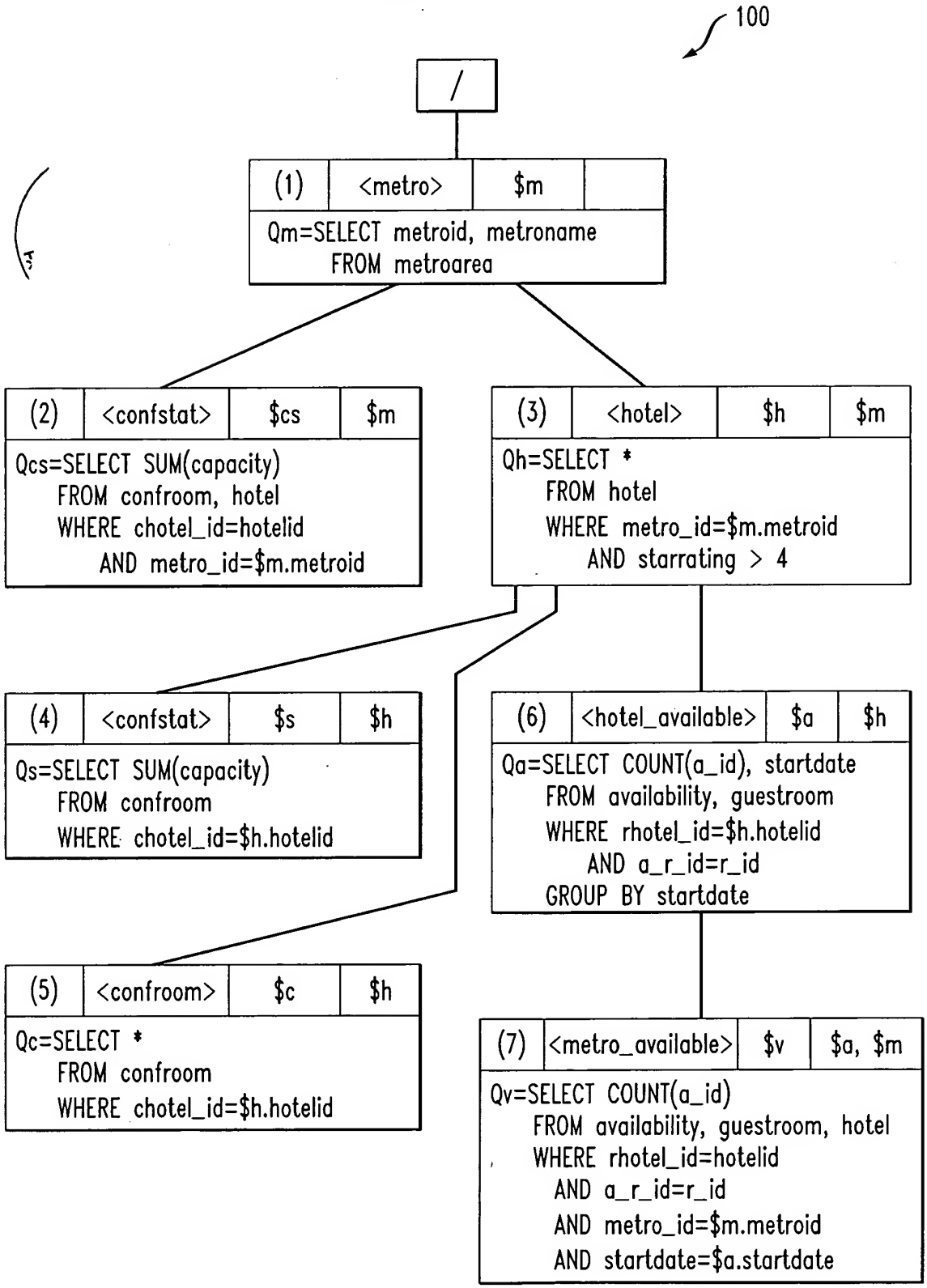

FIG. 2

200

hotelchain(chainid, companyname,hqstate)

metroarea(metroid, metroname)

hotel(hotelid, hotelname, starrating, chain id
metro id, state id, city, pool, gym)

guestroom(r id, rhotel id, roomnumber, type, rackrate)

confroom(c id, chotel id, croomnumber, capacity, rackrate)

availability(a id, a r id, startdate, enddate, price)

FIG. 3

300

```
<metro name="NYCMetroNJ">
    <hotel name="Hyton" pool="Y" gym="N">
        <confstat capacity="3200"/>
        <confroom name="Green Flamingo" capacity="240"/> ...
        ...
    </hotel>
    <hotel> ... </hotel> ...
</metro>
```

FIG. 4

400

```
<xsl:template match="pattern" mode="integer">
    <Result>
        <xsl:apply-templates select="expression"/>
    </Result>
</xsl:template>
```

FIG. 5

```
<xsl:template match="/">                                    (R1)
    <HTML>
        <HEAD></HEAD>
        <BODY>
                <xsl:apply-templates select="metro"/>
        </BODY>
    </HTML>
</xsl:template>
<xsl:template match="metro">                                (R2)
    <result_metro>
      <A></A>
      <xsl:apply-templates
          select="hotel/confstat"/>
    </result_metro>
</xsl:template>
<xsl:template match="confstat">                             (R3)
    <result_confstat>
      <B></B>
      <xsl:apply-templates
        select="../hotel_available/../confroom"/>
    </result_confstat>
</xsl:template>

<xsl:template match="metro/hotel/confroom">                 (R4)
    <xsl:value-of select="."/>
</xsl:template>
```

FIG. 6

600

```
Function PROCESS(x, d_con, mode)
Input: XSLT stylesheet as x, Context document node as d_con,
       Desired mode of the matched rule as mode
Output: XML document fragment as result
1: result ← empty
2: for r_i ∈ x in decreasing order of priority(r_i) do
3:     if mode(r_i) = mode and MATCH(d_con, r_i) then
4:         result ← output (r_i)
5:         for a_j ∈ apply(r_i) do
6:             D_new_con ← SELECT(d_con, a_j)
7:             sub_result ← empty
8:             for d_new_con ∈ D_new_con do
9:                 sub_result ← concatenate(sub_result,
                       PROCESS(x, d_new_con, mode(a_j)))
10:            Replace a_j in result with sub_result
11:        return result
```

FIG. 7

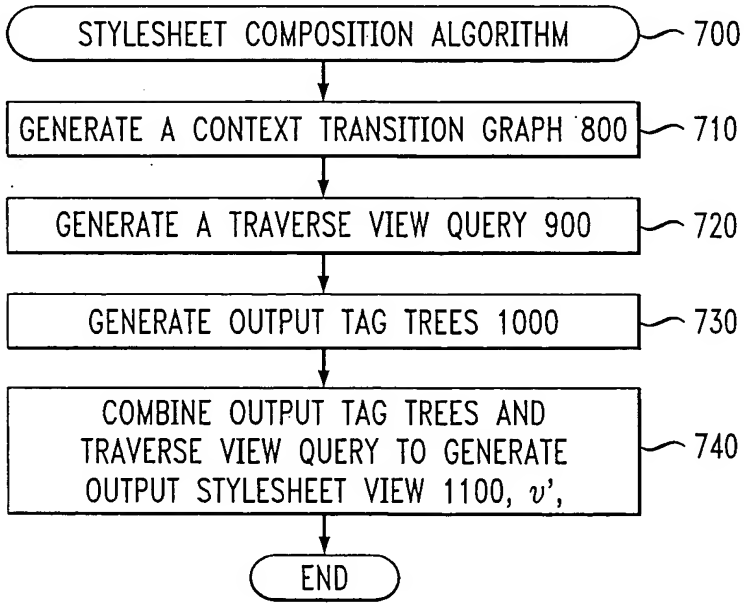

FIG. 8

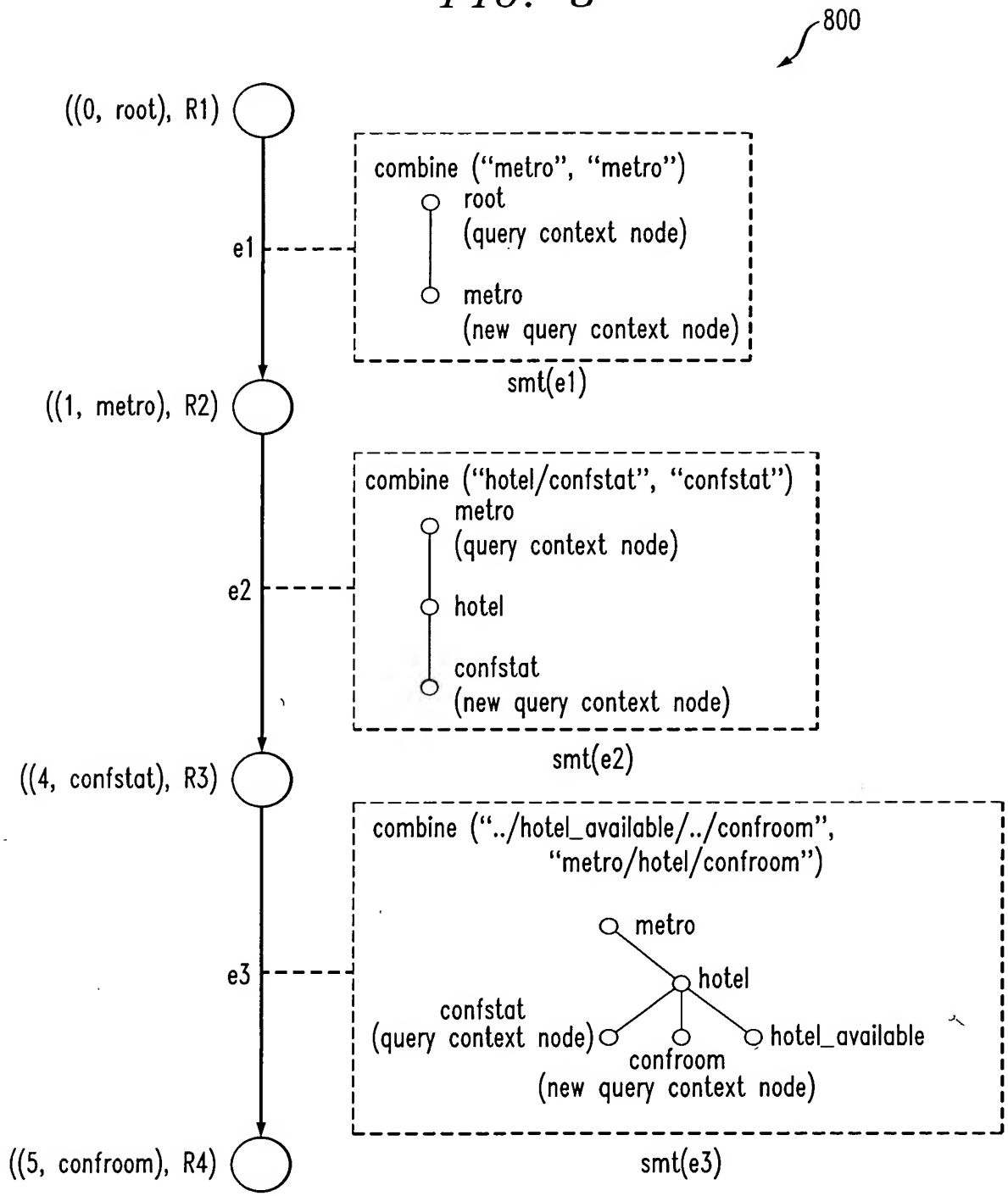
800
((0, root), R1)
combine ("metro", "metro")
root
(query context node)
e1
metro
(new query context node)
smt(e1)
((1, metro), R2)
combine ("hotel/confstat", "confstat")
metro
(query context node)
e2
hotel
confstat
(new query context node)
smt(e2)
((4, confstat), R3)
combine ("../hotel_available/../confroom",
"metro/hotel/confroom")
metro
e3
hotel
confstat
(query context node)
hotel_available
confroom
(new query context node)
smt(e3)
((5, confroom), R4)

FIG. 9

(a) TRAVERSE VIEW QUERY

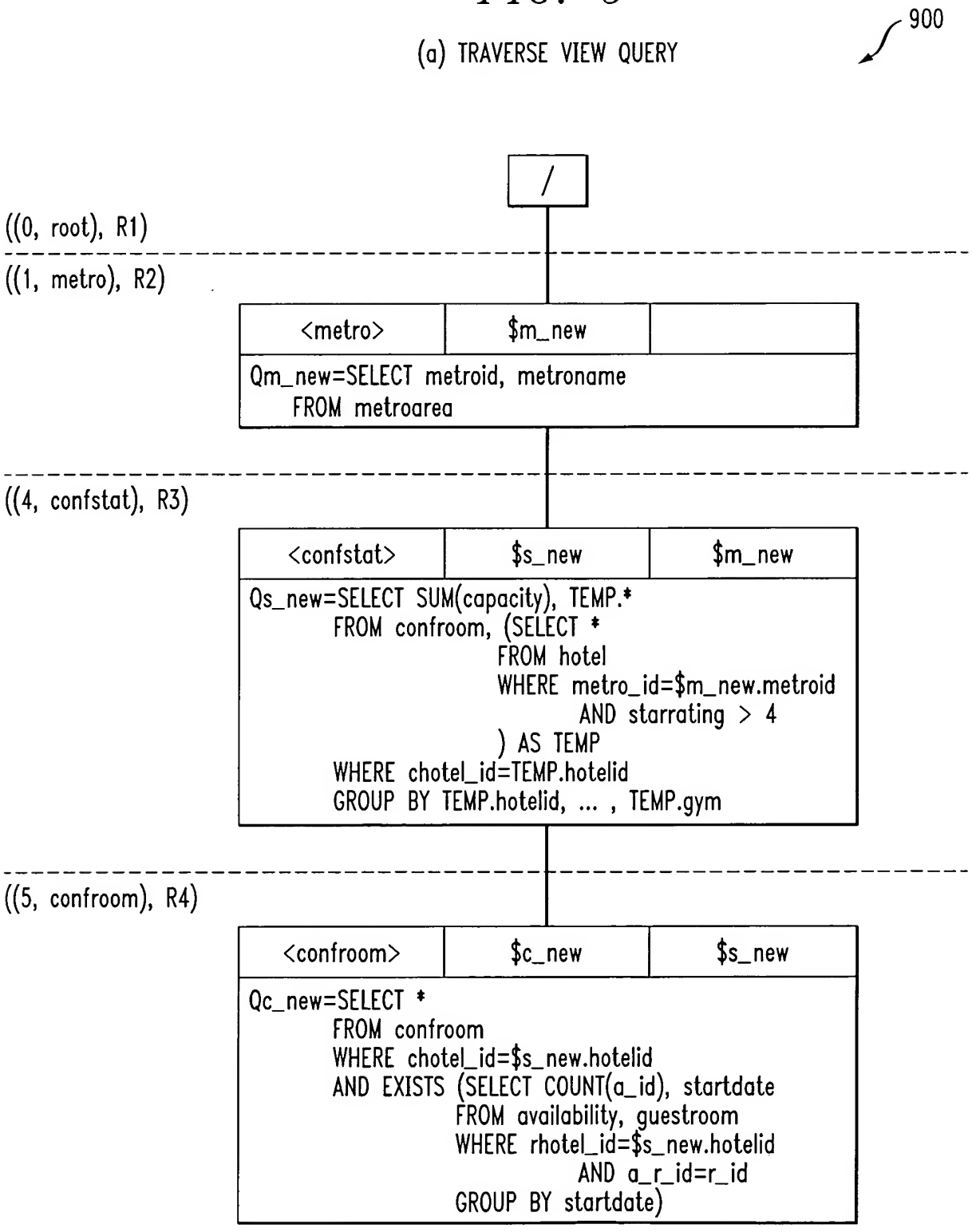

FIG. 10

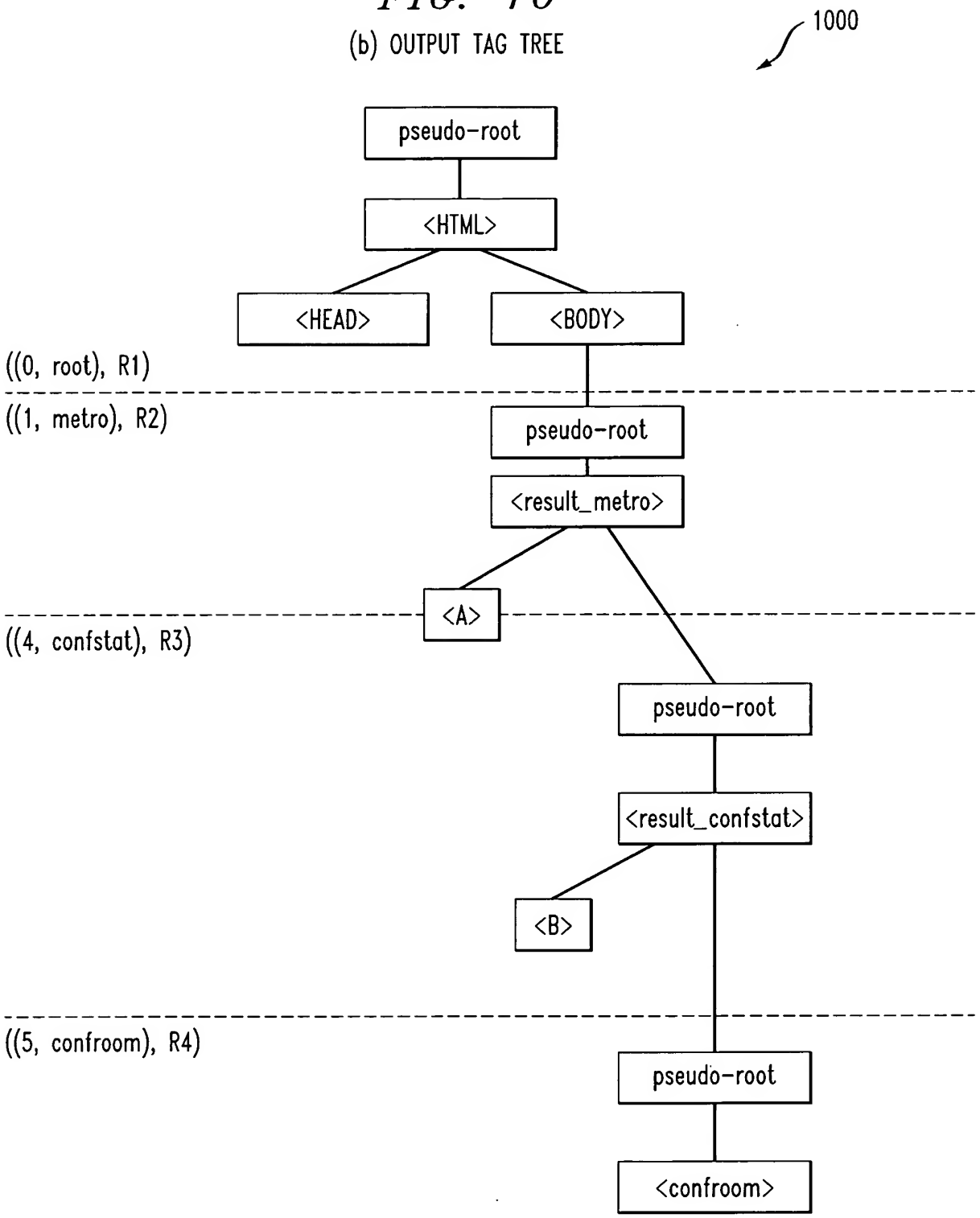
(b) OUTPUT TAG TREE
1000
pseudo-root
<HTML>
<HEAD>
<BODY>
((0, root), R1)
((1, metro), R2)
pseudo-root
<result_metro>
<A>
((4, confstat), R3)
pseudo-root
<result_confstat>
<B>
((5, confroom), R4)
pseudo-root
<confroom>

FIG. 11

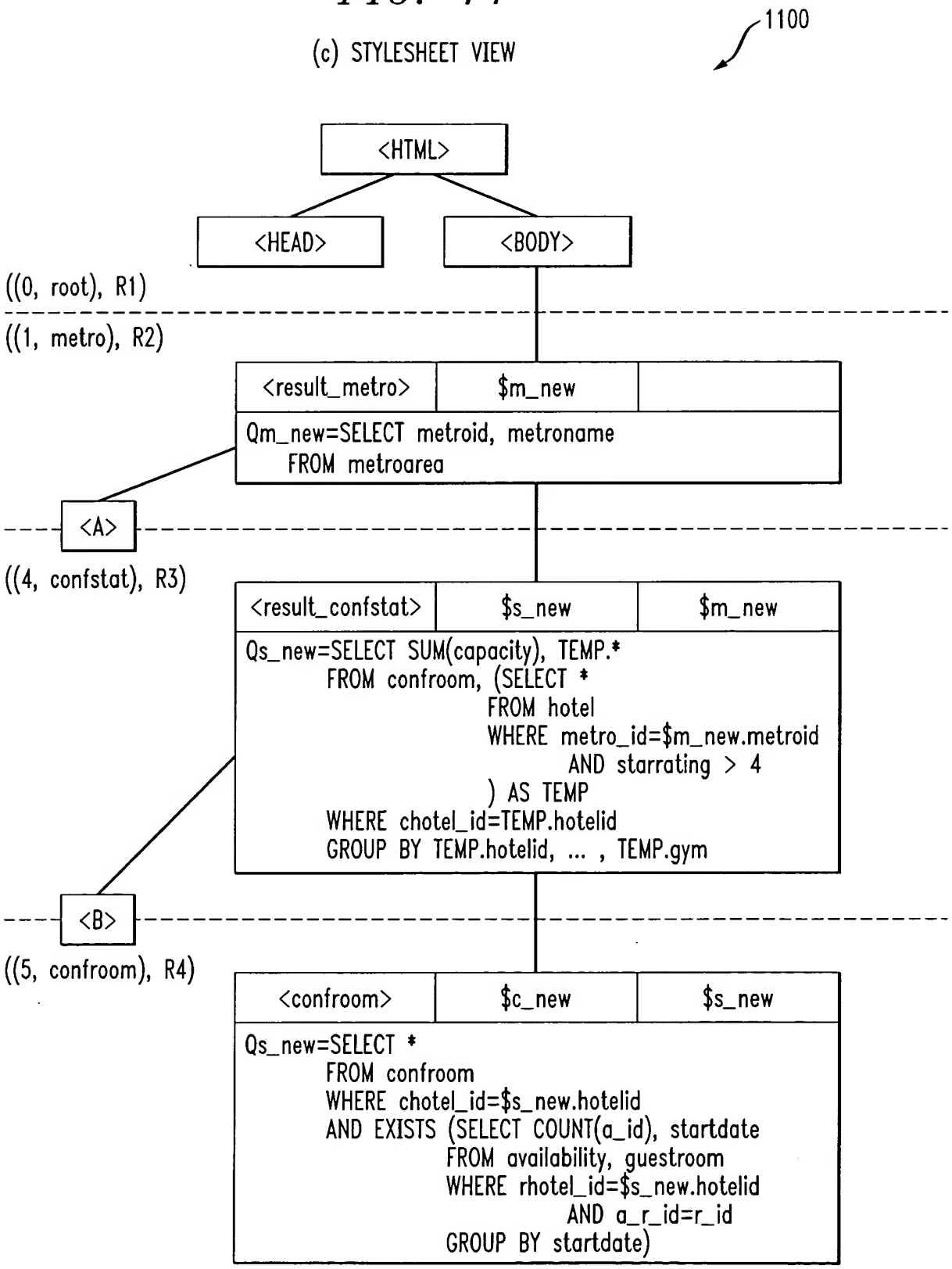
1100
(c) STYLESHEET VIEW
<HTML>
<HEAD>
<BODY>
((0, root), R1)
((1, metro), R2)
<result_metro>
$m_new
Qm_new=SELECT metroid, metroname
FROM metroarea
<A>
((4, confstat), R3)
<result_confstat>
$s_new
$m_new
Qs_new=SELECT SUM(capacity), TEMP.*
FROM confroom, (SELECT *
FROM hotel
WHERE metro_id=$m_new.metroid
AND starrating > 4
) AS TEMP
WHERE chotel_id=TEMP.hotelid
GROUP BY TEMP.hotelid, ... , TEMP.gym
<B>
((5, confroom), R4)
<confroom>
$c_new
$s_new
Qs_new=SELECT *
FROM confroom
WHERE chotel_id=$s_new.hotelid
AND EXISTS (SELECT COUNT(a_id), startdate
FROM availability, guestroom
WHERE rhotel_id=$s_new.hotelid
AND a_r_id=r_id
GROUP BY startdate)

FIG. 12

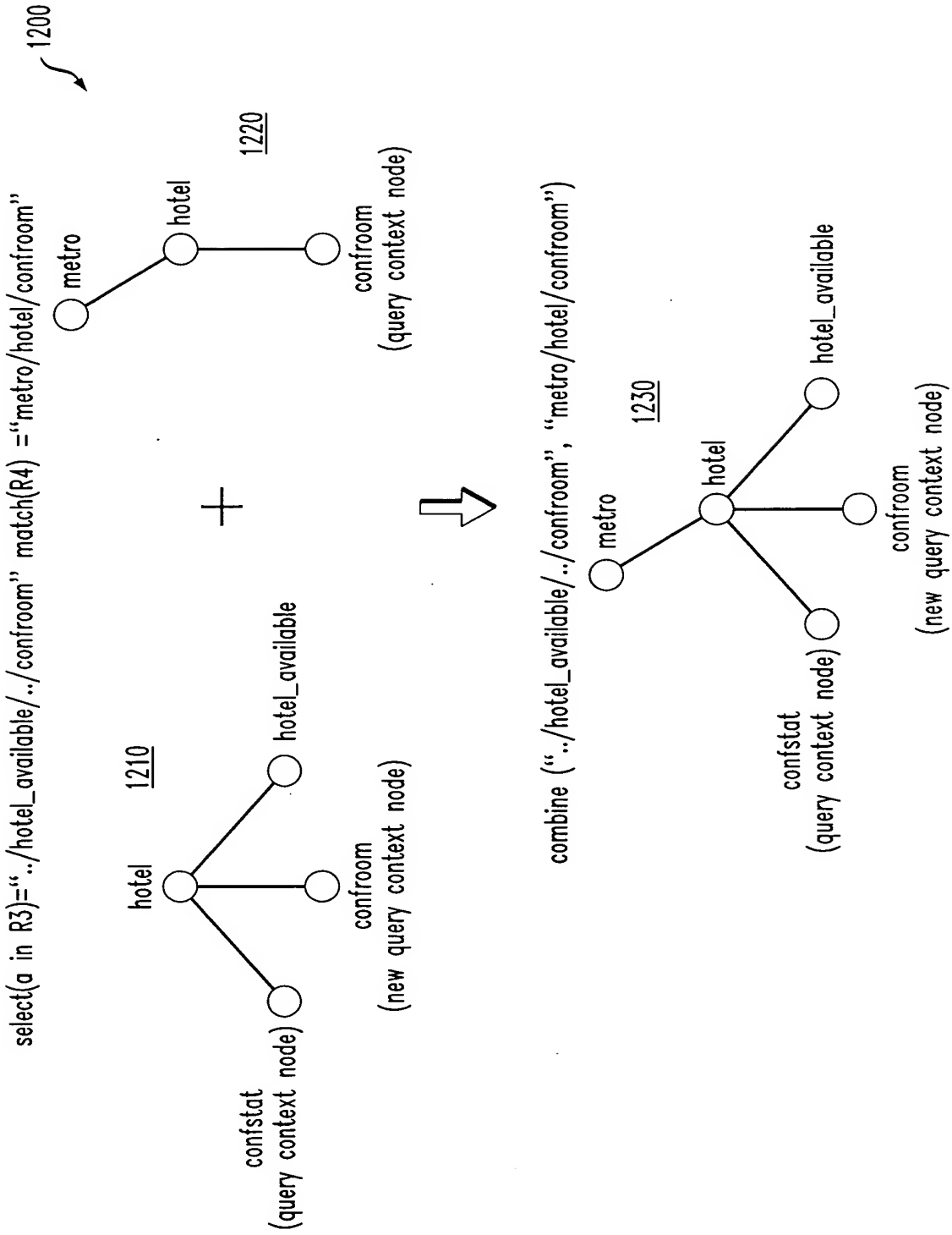
1200
select(a in R3)="../hotel_available/../confroom" match(R4) ="metro/hotel/confroom"
metro
hotel
confroom
(query context node)
1220
hotel
1210
hotel_available
confroom
(new query context node)
confstat
(query context node)
combine ("../hotel_available/../confroom", "metro/hotel/confroom")
1230
metro
hotel
hotel_available
confroom
(new query context node)
confstat
(query context node)

FIG. 13

1300

Procedure *Compose*(v,x)

Input: v: original schema-tree view query; x: XSLT stylesheet

Output: stylesheet view

1: *ctg*: a Context Transition Graph

2: *tvq*: a Traverse View Query

3: *ottree*: an Output Template Tree

4: **for** $n \in v$ **do**

5: **for** $r \in x$ **do**

6: **if** $MATCHQ(n, r) \neq$ NULL **then**

7: add (n, r) to ctg

8: **for** $(n_1, r_1) \in ctg$ **do**

9: **for** $(n_2, r_2) \in ctg$ **do**

10: **for** $a \in apply(r_1)$ **do**

11: $t \leftarrow SELECTQ(n_1, a, n_2)$, $p \leftarrow MATCHQ(n_2, r_2)$

12: **if** $t \neq$ NULL **and** $mode\ (a) = mode(r_2)$ **then**

13: add an edge $e = ((n_1, r_1), (n_2, r_2), a)$ to ctg

14: $smt(e) \leftarrow COMBINE(t, p)$

15: (repeatedly) Delete all nodes without incoming edge, except $(root, r)$

16: $tvq \leftarrow ctg$ {(copy)}, $bvmap$(root of tvq) $\leftarrow$ empty

17: (repeatedly) Duplicate nodes with multiple incoming edges, splitting incoming edges and copying outgoing edges

18: replace binding variables in tvq with new, unique binding variables

19: **for** $e = (w_1 = (n_1, r_1), w_2 = (n_2, r_2), a)$ in edges of ctg **do**

20: $(Q_{bv(w_2)}, bvmap(w_2)) \leftarrow$
$UNBIND(smt(e), n_1, n_2, bv(w_2), bvmap(w_1))$

21: **for** all binding variables bv referenced in $Q_{bv(w_2)}$ **do**

22: rename bv as $bvmap(w_2).get(bv)$

FIG. 13 cont.

```
23: for w = (n,r) ∈ tvq do
24:     ott(w) = GENERATE_OTT(n,r)
25: ottree ← ott(w) ∀ w ∈ tvq {initially a forest}
26: for e = (w1, w2, a) in edges of tvq do
27:         a' ← the "apply-template" node in ott(w1) which is a copy of a
28:         replace a' with an edge from parent(a') to root(ott(w2)))
29: for w = (n, r) ∈ tvq do
30:         bv(root(ott(w))) ← bv(w)
31:         Q_bv(root(ott(w))) ← Q_bv(w)
32: Remove the topmost "pseudo-root" node in ottree
33: while there is any "pseudo-root" node pr left do
34:     for each child node c of pr do
35:         add edge e = (parent(pr), c)
36:         if Q_bv(c) is empty then
37:             bv(c) ← bv(pr), Q_bv(c) ← Q_bv(pr)
38:         else
39:             Q_bv(c) ← UNBIND(parent(pr), c)
40:             add the SELECT columns of Q_bv(pr) to Q_bv(c)
41:             change "bv(pr)" as "bv(c)" in the tag queries of c's
                descendents
42:     remove edge e = (parent(pr), pr) and node pr
43: return ottree {new stylesheet view}
```

FIG. 14

1400

Function *UNBIND*(m, n)
Input: Query context node **as** m, Target node to unbind **as** n
Output: Unbound query for n **as** q
1: $n_j \leftarrow$ the lowest common ancestor of m and n
2: $s_j \leftarrow bv(n_j)$
3: $child_n\ (n_j) \leftarrow$ child node of n_j along the path from n_j to n
4: $s_{j-1} \leftarrow bv(child_n\ (n_j))$
5: $q \leftarrow Q^{s_1, s_2, \ldots, s_{j-1}}_{bv(n)}\ (s_j, \ldots, s_k')$, which is the unbound tag query of n, $Q_{bv(n)}\ (s_1, s_2, \ldots, s_k)$
6: **return** q

FIG. 15

Function *NEST*(p, p')
Input: Node **as** p, Child node **as** p'
Output: Tag query for p after nesting **as** q
1: $q \leftarrow Q_{bv(p)}$
2: **for** each child node c of p, except p' **do**
3: $\quad q_c \leftarrow NEST(c, NULL)$
4: $\quad$ add *EXISTS* q_c in *WHERE* clause of q
5: **return** q

FIG. 16

1: {**Replace line 5 of Figure 10**}
2: $P \leftarrow$ {nodes along the path from $child_n\ (n_j)$ to n}
3: **for** node $p \in P$ **do**
4: $\quad p' \leftarrow$ child of $p \in P$, otherwise *NULL*
5: $\quad \Theta_{bv(p)} \leftarrow NEST(p, p')$
6: $q \leftarrow \Theta^{s_1, s_2, \ldots, s_{j-1}}_{bv(n)}\ (s_j, \ldots, s'_k)$, which is the unbound and nested tag query of n, $\Theta_{bv(n)}\ (s_1, s_2, \ldots, s_k)$, and decorrelation of $bv(s_i)$ is done by Θ_{si}.

FIG. 17

Function *UNBIND*(*smt*, *m*, *n*, *bv'*, *bvmap*)
Input: Select–match subtree **as** *smt*, Query context node of *smt* **as** *m*, New query context node of *smt* **as** *n*, New binding variable as *bv'*, Binding variable map **as** *bvmap*.
Output: Unbound query for *smt* **as** *q*, New binding variable map **as** *bvmap'*.

1: $q \leftarrow UNBIND(m, n)$
2: $n_j \leftarrow$ the lowest common ancestor of m and n
3: $child_n\ (n_j) \leftarrow$ child node of n_j along the path from n_j to n
4: $R \leftarrow$ {nodes along the path from $child_n\ (n_j)$ to n}
5: **for** node $p \in R$ **do**
6: add the *SELECT* columns of $Q_{bv(p)}$ to q
7: $P \leftarrow$ {nodes along the path from root of *smt* to m}
8: **for** node $p \in P$ **do**
9: **for** each child node c of p, such that $c \notin P$ and $c \notin R$ **do**
10: $q_c \leftarrow NEST(c, NULL)$
11: add *EXISTS* q_c in *WHERE* clause of q
12: $bvmap' \leftarrow bvmap$
13: **for** node $p \in R$ **do**
14: $bvmap'.insert(bv(p), bv')$
15: $child_m\ (n_j) \leftarrow$ child node of n_j along the path from n_j to m
16: $S \leftarrow$ {nodes along the path from $child_m\ (n_j)$ to m}
17: **for** node $s \in S$ **do**
18: $bvmap'.remove(bv(s))$
19: **return** $(q, bvmap')$

FIG. 18

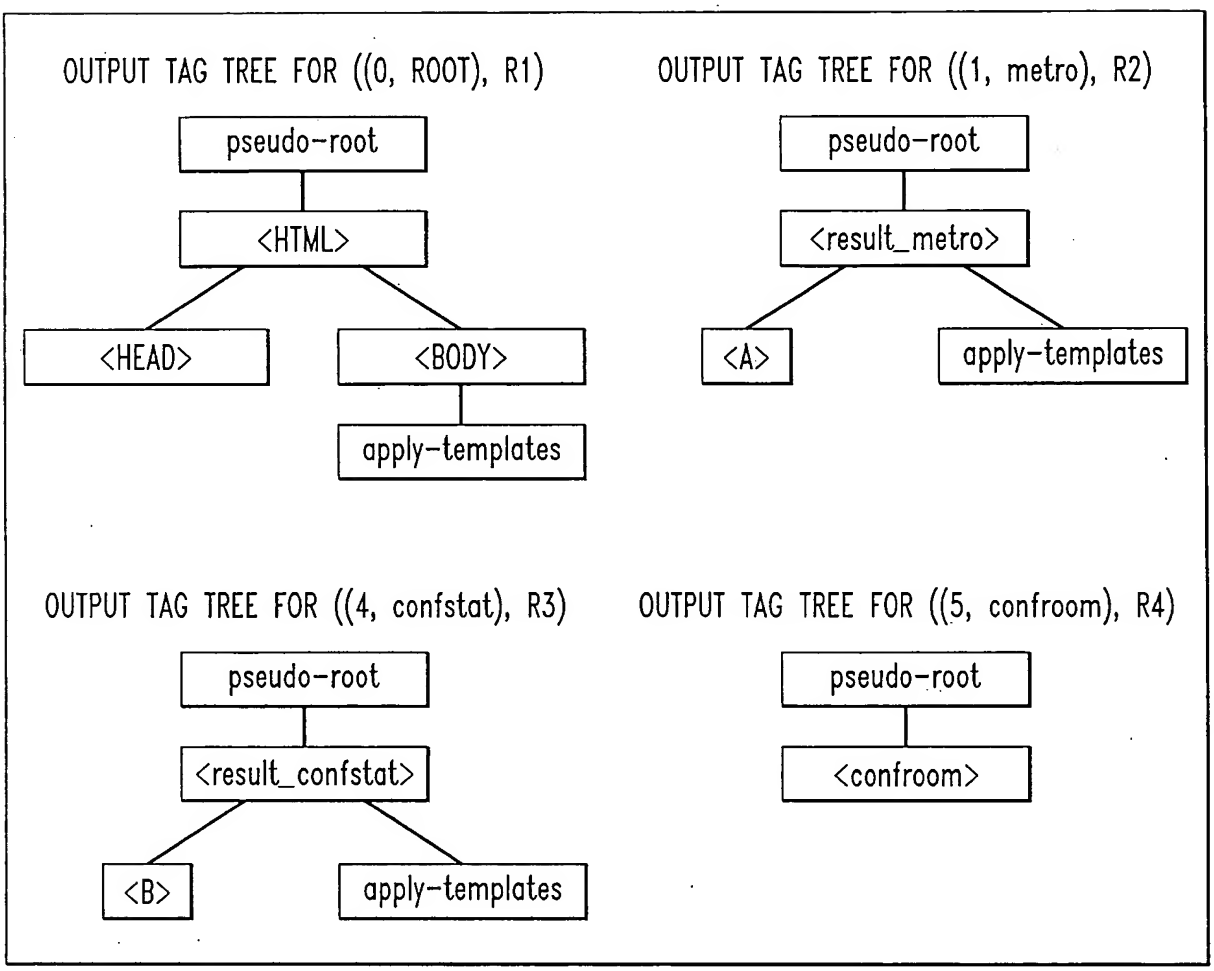
OUTPUT TAG TREE FOR ((0, ROOT), R1)
pseudo-root
<HTML>
<HEAD>
<BODY>
apply-templates
OUTPUT TAG TREE FOR ((1, metro), R2)
pseudo-root
<result_metro>
<A>
apply-templates
OUTPUT TAG TREE FOR ((4, confstat), R3)
pseudo-root
<result_confstat>
<B>
apply-templates
OUTPUT TAG TREE FOR ((5, confroom), R4)
pseudo-root
<confroom>

FIG. 19

```
(R1), (R3), and (R4) are the same as Figure 4.

<xsl:template match="metro">                                    (R2)
    <xsl:apply-templates select="hotel/confstat"/>
</xsl:template>
```

FIG. 20

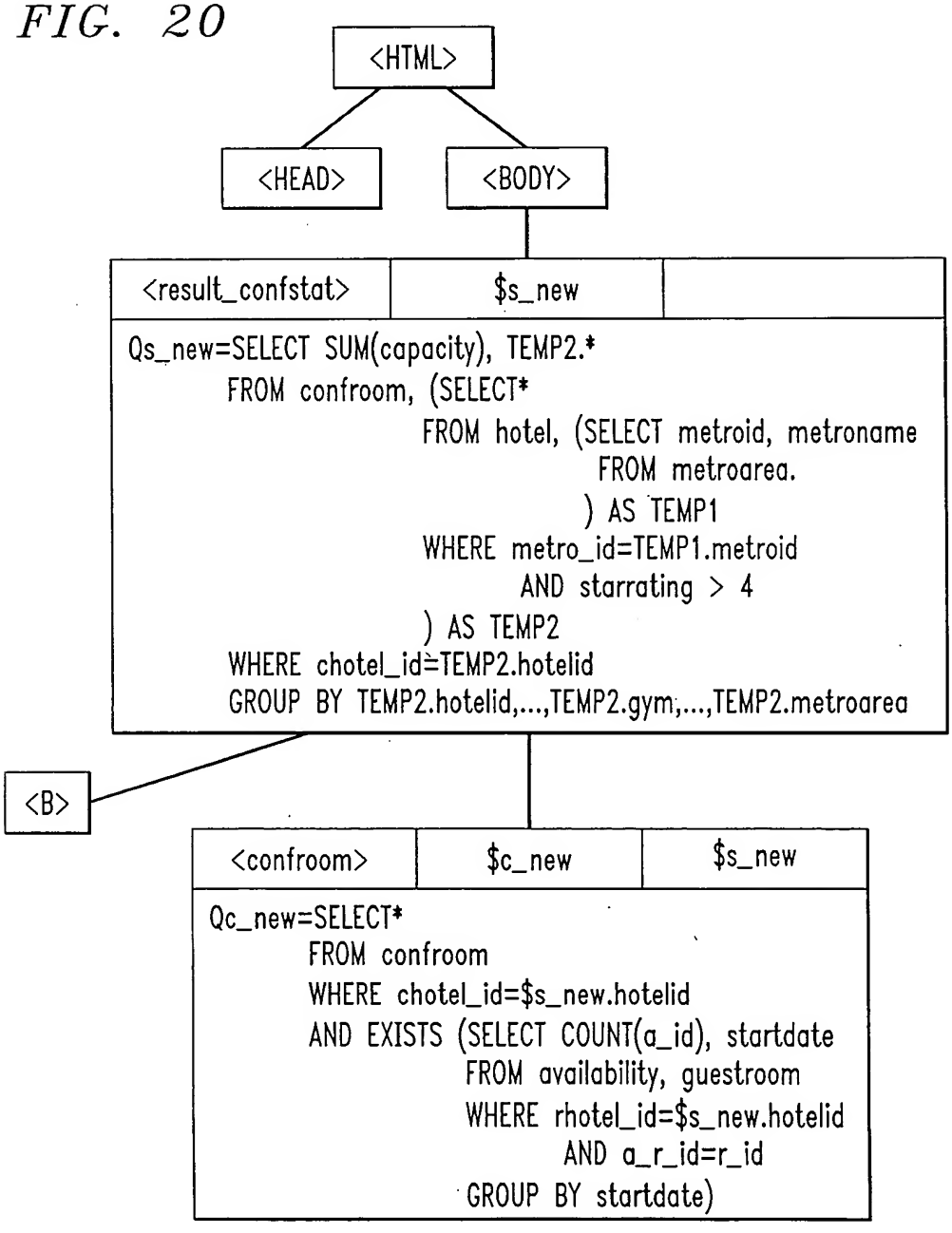

FIG. 21

```
(R1) and (R2) are the same as Figure 4.

<xsl:template match="confstat">                              (R3)
    <result_confstat>
        <B/>
        <xsl:apply-templates select=".[@sum<200]/
            ../hotel_available/../confroom
            [../confstat[@sum>100]] [@capacity>250]"/>
    </result_confstat>
</xsl:template>

<xsl:template match="metro[@metroname=
                "chicago"]/hotel/confroom">                  (R4)
        <xsl:value-of select="."/>
</xsl:template>
```

FIG. 22

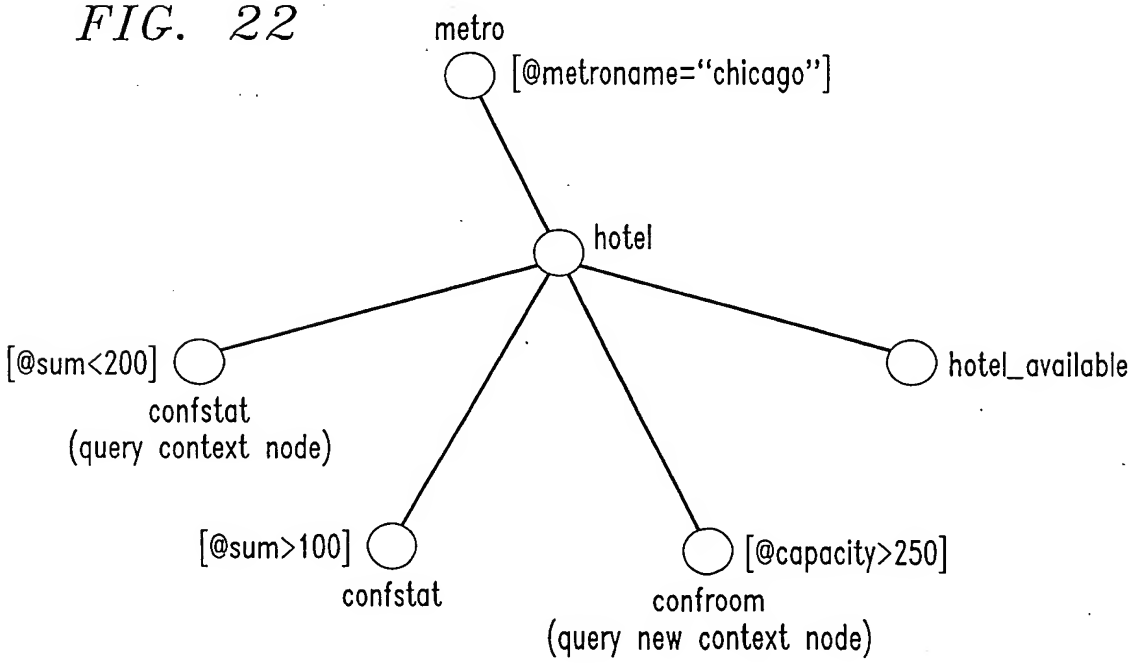

FIG. 23

{Insert the following
(a) after line 8 of Figure 13
(b) after line 1 of Figure 11}
$e_p \leftarrow$ predicate using $tag(p)$
add e_p in *WHERE* clause of q

FIG. 24

```
SELECT * FROM confroom
WHERE chotel_id=$s_new.hotelid
      AND capacity > 250
      AND $s_new.SUM_capacity<200
      AND $s_new.metroname="chicago"
      AND EXISTS (SELECT SUM (capacity)
                  FROM confroom,
                  WHERE chotel_id=$s_new.hotelid
                  HAVING SUM (capacity) > 100)
      AND EXISTS (SELECT COUNT (a_id), startdate
                  FROM availability, guestroom
                  WHERE rhotel_id=$s_new.hotelid
                        AND a_r_id=r_id
                  GROUP BY startdate)
```

FIG. 25

(a)

```
<xsl:template match="pattern" mode="m">
    <xsl:if test="expression">
        template body
    </xsl:if>
</xsl:template>
```

(b)

```
<xsl:template match="pattern" mode="m">
    <xsl:apply-templates select=" . [expression] " mode="mnew"/>
</xsl:template>

<xsl:template match="nodename" mode="mnew">
    template body
</xsl:template>
```

FIG. 26

(a)

```
<xsl:template match="pattern" mode="m">
    <xsl:choose>
        <xsl:when test="e1">b1</xsl:when>
        <xsl:when test="e2">b2</xsl:when>
        <xsl:otherwise>b3</xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

(b)

```
<xsl:template match="pattern" mode="m">
    <xsl:apply-templates
        select=" . [e1] " mode="mnew1"/>
    <xsl:apply-templates
        select=" . [not (e1) and e2] "
        mode="mnew2"/>
    <xsl:apply-templates
        select=" . [not (e1) and not (e2)] "
        mode="mnew3"/>
</xsl:template>

<xsl:template match="nodename"
        mode="mnew1">b1</xsl:template>
<xsl:template match="nodename"
        mode="mnew2">b2</xsl:template>
<xsl:template match="nodename"
        mode="mnew3">b3</xsl:template>
```

FIG. 27

(a)

```
<xsl:template match="pattern" mode="m">
    <xsl:value-of select="expression"/>
</xsl:template>
```

(b)

```
<xsl:template match="pattern" mode="m">
    <xsl:apply-templates
        select="path expression" mode="mnew"/>
</xsl:template>

<xsl:template match="nodename[predicate]"
        mode="mnew">
    <xsl:value-of select=" . "/>
</xsl:template>
```

FIG. 28

(a)

```
<xsl:template match="pattern 1" mode="m">
    template body 1
</xsl:template>

<xsl:template match="pattern 2" mode="m">
    template body 2
</xsl:template>
```

(b)

```
<xsl:template match="pattern 1" mode="m1">
    template body 1
</xsl:template>

<xsl:template match="pattern 2" mode="m">
    <xsl:choose>
        <xsl:when test="expression 1">
            <xsl:apply-templates select=" . "
                    mode="m1"/>
        </xsl:when>
        <xsl:otherwise>
            template body 2
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

FIG. 29

```
<xsl:template match="/metro">                                    (R1)
    <xsl:param name="idx" select="10"/>
    <result_metro>
        <xsl:apply-templates
        select="hotel/hotel_available[@count>10]
                /metro_available[@count<$idx] ">
        <xsl:with-param name="idx"
                select="$idx"/>
        </xsl:apply-templates>
    </result_metro>
</xsl:template>

<xsl:template match="metro_available">                           (R2)
    <xsl:param name="idx"/>
    <xsl:choose>
        <xsl:when test="idx<=1">
            <xsl:value-of select=" . "/>
        </xsl:when>
        <xsl:otherwise>
            <result_metroavail>
                <xsl:apply-templates
                        select="self::[@count>50]/../../..">
                    <xsl:with-param name="idx"
                                    select="$idx-1"/>
                </xsl:apply-templates>
            <result_metroavail>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

FIG. 30

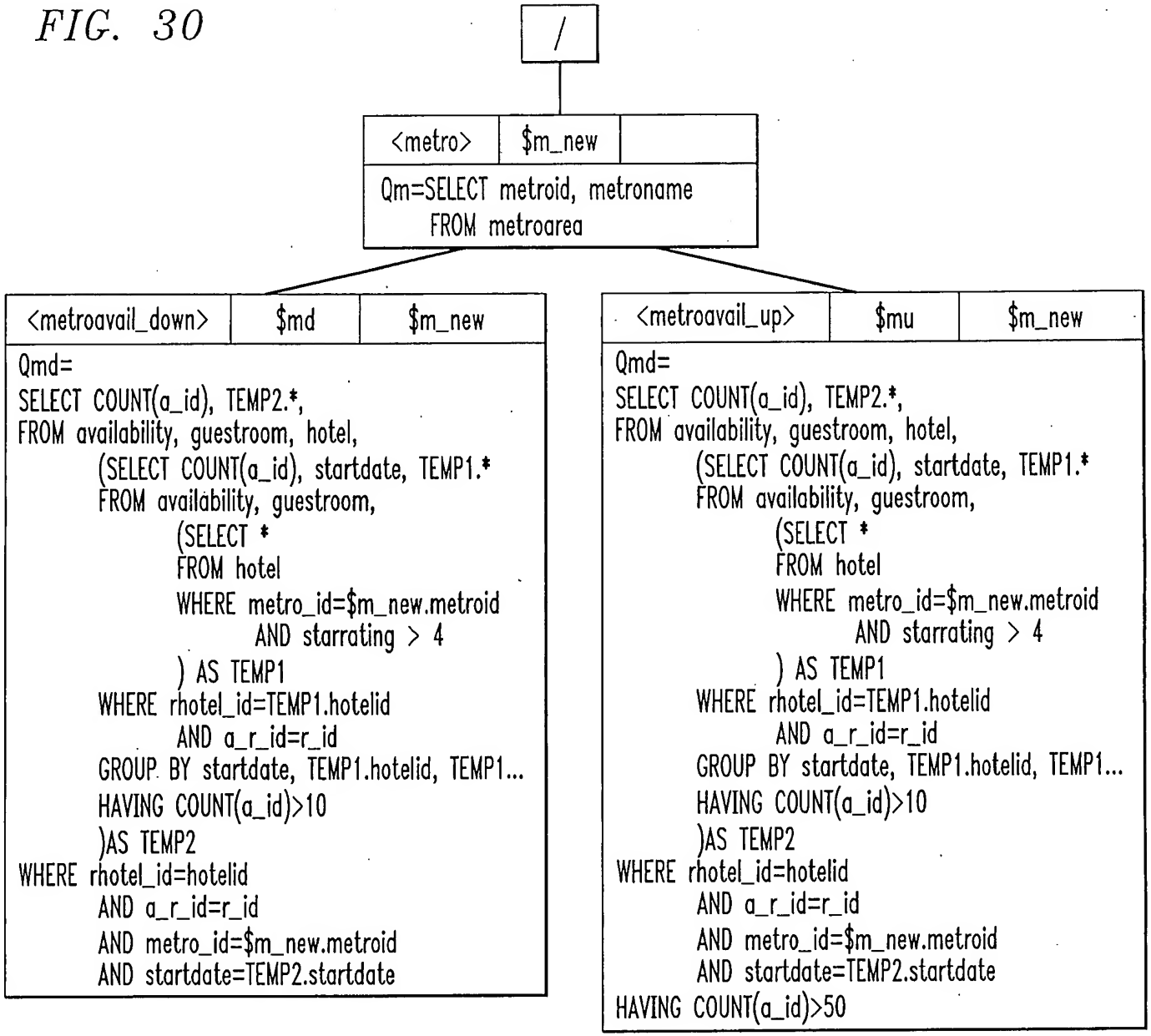
/
<metro>
$m_new
Qm=SELECT metroid, metroname
FROM metroarea
<metroavail_down>
$md
$m_new
Qmd=
SELECT COUNT(a_id), TEMP2.*,
FROM availability, guestroom, hotel,
(SELECT COUNT(a_id), startdate, TEMP1.*
FROM availability, guestroom,
(SELECT *
FROM hotel
WHERE metro_id=$m_new.metroid
AND starrating > 4
) AS TEMP1
WHERE rhotel_id=TEMP1.hotelid
AND a_r_id=r_id
GROUP BY startdate, TEMP1.hotelid, TEMP1...
HAVING COUNT(a_id)>10
)AS TEMP2
WHERE rhotel_id=hotelid
AND a_r_id=r_id
AND metro_id=$m_new.metroid
AND startdate=TEMP2.startdate
<metroavail_up>
$mu
$m_new
Qmd=
SELECT COUNT(a_id), TEMP2.*,
FROM availability, guestroom, hotel,
(SELECT COUNT(a_id), startdate, TEMP1.*
FROM availability, guestroom,
(SELECT *
FROM hotel
WHERE metro_id=$m_new.metroid
AND starrating > 4
) AS TEMP1
WHERE rhotel_id=TEMP1.hotelid
AND a_r_id=r_id
GROUP BY startdate, TEMP1.hotelid, TEMP1...
HAVING COUNT(a_id)>10
)AS TEMP2
WHERE rhotel_id=hotelid
AND a_r_id=r_id
AND metro_id=$m_new.metroid
AND startdate=TEMP2.startdate
HAVING COUNT(a_id)>50

FIG. 31

```
<xsl:template match="/metro">                                    (R1')
      <xsl:param name="idx" select="10"/>
      <result_metro>
            <xsl:apply-templates
                  select="metroavail_down[@count<$idx]">
               <xsl:with-param name="idx"
                  select="$idx"/>
            </xsl:apply-templates>
      </result_metro>
</xsl:template>

<xsl:template match="metroavail_down">                           (R2')
      <xsl:param name="idx"/>
      <xsl:choose>
            <xsl:when test="$idx<=1">
               <xsl:value-of select="."/>
            </xsl:when>
            <xsl:otherwise>
                  <result_metroavail>
                     <xsl:apply-templates
                           select="../metroavail_up">
                        <xsl:with-param
                           name="idx" select="$idx-1"/>
                     </xsl:apply-templates>
                  <result_metroavail>
            </xsl:otherwise>
      </xsl:choose>
</xsl:template>

<xsl:template match="metroavail_up">                             (R3')
      <xsl:param name="idx"/>
            <result_metro>
                  <xsl:apply-templates
                        select="../metroavail_down[@count<$idx]">
                        <xsl:with-param
                           name="idx" select="$idx"/>
                  </xsl:apply-templates>
            <result_metro>
</xsl:template>
```